ESTRUTURA E RECUPERAÇÃO DE DADOS A

INDEXADOR E CONTADOR DE PALAVRAS

Grupo: máximo de 4 integrantes

PROJETO BÁSICO (5 pontos)

Este projeto consiste na implementação de um sistema para indexação e contagem de palavras de um texto. Durante o desenvolvimento deste projeto, você se familiarizará com a noção de *tabela de símbolos*. A referência básica para esta parte do projeto é o Capítulo 3 do livro Algorithms do autor Robert Sedgewick (http://algs4.cs.princeton.edu/30searching).

Problema a ser resolvido

O programa que você escreverá neste projeto deve resolver o seguinte problema: a entrada de seu programa é formada por um inteiro **n** e um **texto** e a saída de seu programa deve ser a lista das **n** palavras mais frequentes no texto com suas respectivas frequências (número de ocorrências), em ordem decrescente de frequência, sendo exibidas em cada linha a frequência e a palavra, nesta ordem, separadas por um espaço em branco.

Palavras

Uma *palavra* é uma sequência maximal de letras. Você deve distinguir letras maiúsculas de minúsculas, de forma que “Vitória” e “vitória” devem ser consideradas palavras distintas. As palavras que ocorrem um mesmo número de vezes devem ser listadas em ordem alfabética (considere a ordem da tabela ASCII). O seu programa deve parar após **n** palavras terem sido exibidas, caso a entrada tenha pelo menos **n** palavras. Caso a entrada tenha menos de **n** palavras, seu programa deve parar após exibir todas as palavras existentes.

Execução

O seu programa deve receber um inteiro **n** na linha de comando, através da opção **-nNUMERO**, onde **NUMERO** deve ser substituído pelo número de palavras desejado, e o texto de entrada de seu programa deve vir da **entrada padrão (stdin)**. A saída deve ser enviada para a **saída padrão (stdout)**.

Executando o seu programa com ***n*** igual a **5** e o arquivo **“tale.txt**” (http://algs4.cs.princeton.edu/31elementary/tale.txt) como entrada, obtemos como saída:

```
$ ./projeto2  -n5  < tale.txt
7989 the
4931 and
4002 of
3460 to
2909 a
```

Esta saída diz que as 5 palavras mais frequentes no arquivo “tale.txt” são as palavras “the”, “and”, “of”, “to” e “a”, nesta ordem. A palavra mais frequente é “the”, com 7989 ocorrências, seguida pela palavra “and” com 4931 ocorrências e assim por diante.

Por outro lado, executando o seu programa com ***n*** igual a **10** e o arquivo "**tinyTale.txt**" (http://algs4.cs.princeton.edu/31elementary/tinyTale.txt) como entrada, obtemos como saída:

```
$ ./projeto2  -n10  < tinyTale.txt
10 it
10 of
10 the
10 was
2 age
2 epoch
2 season
2 times
1 belief
1 best
```

Esta saída diz que as 10 palavras mais frequentes no arquivo "tinyTale.txt" são as palavras "it", "of", "the", "was", "age", "epoch", "season", "times", "belief" e "best", nesta ordem. A palavra mais frequente é "it", com 10 ocorrências, seguida pela palavra "of", também com 10 ocorrências e assim por diante. Repare que palavras com um mesmo número de ocorrências são exibidas em ordem alfabética.

Objetos da tabela de símbolos

Os objetos a serem armazenados em sua tabela de símbolos devem ser do tipo **Item**, implementados no arquivo **Item.c** e manipulados através da interface **Item.h**.

Tabela de símbolos

Uma *tabela de símbolos* é uma estrutura de dados de **Items** compostos por uma **Chave** e um **Valor** associado a esta chave, que permitem dois tipos de operações básicas: inserir um novo **Item** e devolver um **Item** que possui uma determinada chave.

Tabelas de símbolos são também chamadas de dicionários devido à analogia com o sistema que fornece o significado de palavras listando-as em ordem alfabética. Neste exemplo temos que a **Chave** é uma palavra e o **Valor** é o significado desta palavra. Em um programa em linguagem C, para este exemplo, um **Item** poderia ser declarado da seguinte forma:

```
struct tipoItem {
   char chave[50];      /* string que contem a palavra do dicionario */
   char valor[200];     /* string contendo o significado desta palavra */
};
typedef struct tipoItem Item;
```

Para implementar a sua tabela de símbolos você deverá utilizar uma **árvore binária de busca (ABB)**. A sua tabela de símbolos deve ser implementada no arquivo **ST.c** e o acesso a ela deve ser *estritamente através da interface* **ST.h**. Projete a interface **ST.h** de forma que ao alterar a implementação da tabela de símbolos não seja necessário alterar outras partes do programa.

OPCIONAL 1 (até 1 ponto)

O programa deve implementar uma opção adicional **-wFILENAME**, que permitirá salvar as informações da tabela de símbolos em um arquivo, cujo nome deve ser informado no lugar de **FILENAME**, e uma opção adicional **-rFILENAME**, que permitirá carregar as informações da tabela de símbolos de um arquivo, cujo nome deve ser informado no lugar de **FILENAME**.
Executando o seu programa com ***n*** igual a **5**, **w** igual a "**stfile**" e o arquivo "**tale.txt**" como entrada, teremos como resultado o conteúdo da tabela de símbolos salvo no arquivo "**stfile**", obtendo como saída:

$ ./projeto2 -n5 -wstfile < tale.txt
7989 the
4931 and
4002 of
3460 to
2909 a

Por outro lado, executando o seu programa com ***n*** igual a **5** e **r** igual a "**stfile**", o arquivo de entrada não deverá ser informado, sendo o conteúdo da tabela de símbolos carregado do arquivo "**stfile**", obtendo como saída:

$ ./projeto2 -n5 -rstfile
7989 the
4931 and
4002 of
3460 to
2909 a

Caso as opções **w** e **r** sejam informadas simultaneamente, apenas a opção **w** deverá ser executada.

OPCIONAL 2 (até 1 ponto)

O programa deve implementar uma opção adicional **-sWORD**, que permitirá buscar na tabela de símbolos pela palavra informada no lugar de **WORD**, exibindo uma linha contendo a palavra, sua frequência, sua altura na árvore binária de busca (considere que a raiz da árvore possui altura igual a 1) e o tempo de busca (em microsegundos), separados por espaço.

OPCIONAL 3 (até 2 pontos)

O programa deve implementar uma opção adicional **-b**, que fará com que a tabela de símbolos utilizada seja uma **árvore AVL (árvore binária de busca balanceada)**, implementada no arquivo **STb.c**, com acesso *estritamente através da interface* **STb.h**. Projete a interface **STb.h** de forma que ao alterar a implementação da tabela de símbolos não seja necessário alterar outras partes do programa.

OPCIONAL 4 (até 2 pontos)

O programa deve implementar uma opção adicional **-pMAX**, que fará com que as palavras existentes na tabela de símbolos sejam exibidas na forma de uma árvore, até a altura máxima informada no lugar de **MAX**. Caso algum dos filhos não exista, a

palavra **NULL** deve ser exibida em seu lugar. Com o valor de **MAX** igual a **4**, por exemplo, teríamos a estrutura mostrada a seguir:

```
raiz
|- filho da esquerda (altura 2)
|  |- filho da esquerda (altura 3)
|  |  |- filho da esquerda (altura 4)
|  |  |- filho da direita (altura 4)
|  |- filho da direita (altura 3)
|     |- filho da esquerda (altura 4)
|     |- filho da direita (altura 4)
|- filho da direita (altura 2)
   |- filho da esquerda (altura 3)
   |  |- filho da esquerda (altura 4)
   |  |- filho da direita (altura 4)
   |- filho da direita (altura 3)
      |- filho da esquerda (altura 4)
      |- filho da direita (altura 4)
```

OBSERVAÇÕES

1. No início do arquivo que contiver o programa principal, deve haver o seguinte cabeçalho preenchido pelo grupo:

Integrante 1 - Nome: _____ RA: _____
Integrante 2 - Nome: _____ RA: _____
Integrante 3 - Nome: _____ RA: _____
Integrante 4 - Nome: _____ RA: _____
Resultados obtidos: _____
Projeto básico: _____ % concluído - Obs: _____
( ) Opcional 1 - Obs: _____
( ) Opcional 2 - Obs: _____
( ) Opcional 3 - Obs: _____
( ) Opcional 4 - Obs: _____

2. O trabalho vale até 10 pontos. Os grupos podem implementar quantos opcionais desejarem, mas a nota do projeto será limitada a 10 pontos. Dessa forma, cada grupo deve escolher um conjunto de opcionais, cujos pontos serão somados aos 5 pontos do projeto básico (obrigatório). Cada grupo tem a liberdade de escolher os opcionais que considerar mais convenientes.

3. O trabalho deverá ser elaborado em linguagem C, em ambiente Linux, e deve ser criado um Makefile para a compilação do seu programa.

4. Qualquer tentativa de plágio será punida com a nota **-Nmax** para todos os integrantes do grupo.

5. Cada membro do grupo deve postar em seu escaninho no AVA, na pasta "**Projeto2**" o código fonte produzido. Não serão aceitos trabalhos postados após a data de entrega.